Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!


## Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China

# P5Q Serial Phase Change Memory (PCM)

## Features

- SPI bus compatible serial interface
- Maximum clock frequency
  - 66 MHz (0°C to +70°C)
  - 33 MHz (–30°C to +85°C)
- 2.7V to 3.6V single supply voltage
- Supports legacy SPI protocol and new quad I/O or dual I/O SPI protocol
- Quad I/O frequency of 50 MHz, resulting in an equivalent clock frequency up to 200 MHz
- Dual I/O frequency of 66 MHz, resulting in an equivalent clock frequency up to 132 MHz
- Continuous READ of entire memory via single instruction:
  - Quad and dual output fast read
  - Quad and dual input fast program
- Uniform 128Kb sectors (Flash emulation)
- WRITE operations
  - 128Kb sectors ERASE (emulated)
  - Legacy Flash PAGE PROGRAM
  - Bit-alterable page WRITEs
  - PAGE PROGRAM on all 1s (PRESET WRITEs)
- Write protections: protected area size defined by four nonvolatile bits (BP0, BP1, BP2, and BP3)
- JEDEC-standard two-byte signature (DA18h)
- 128Mb density with SOIC16 package
- More than 1,000,000 WRITE cycles
- Phase change memory (PCM)
  - Chalcogenide phase change storage element
  - Bit-alterable WRITE operation

## Table of Contents

## List of Figures

# List of Tables

# Functional Description

P5Q serial phase change memory (PCM) is nonvolatile memory that stores information through a reversible structural phase change in a chalcogenide material. The material exhibits a change in material properties, both electrical and optical, when changed from the amorphous (disordered) to the polycrystalline (regularly ordered) state. In the case of PCM, information is stored via the change in resistance that the chalcogenide material experiences when undergoing a phase change. The material also changes optical properties after experiencing a phase change, a characteristic that has been successfully mastered for use in current rewritable optical storage devices, such as rewritable CDs and DVDs.

The P5Q serial PCM storage element consists of a thin film of chalcogenide contacted by a resistive heating element. In PCM, the phase change is induced in the memory cell by highly localized Joule heating caused by an induced current at the material junction. During a WRITE operation, a small volume of the chalcogenide material is made to change phase. The phase change is a reversible process and is modulated by the magnitude of injected current, the applied voltage, and the duration of the heating pulse.

Unlike other proposed alternative memories, P5Q serial PCM technology uses a conventional CMOS process with the addition of a few additional layers to form the memory storage element. Overall, the basic memory manufacturing process used to make PCM is less complex than that of NAND, NOR, or DRAM.

P5Q serial PCM combines the benefits of traditional floating gate Flash, both NOR-type and NAND-type, with some of the key attributes of RAM and EEPROM. Like NOR Flash and RAM technology, PCM offers fast random access times. Like NAND Flash, PCM has the ability to write moderately fast, and like RAM and EEPROM, PCM supports bit-alterable WRITEs (overwrite). Unlike Flash, no separate erase step is required to change information from 0 to 1 and 1 to 0. Unlike RAM, however, the technology is nonvolatile with data retention compared with NOR Flash.

# Product Features

P5Q serial PCM devices have 128Mb (16Mb x 8Mb) SPI phase change memory with advanced write protection mechanisms, accessed by a high-speed, SPI-compatible bus. The memory can be programmed from 1 to 64 bytes at a time using the PAGE PROGRAM, DUAL INPUT FAST PROGRAM, and QUAD INPUT FAST PROGRAM instructions. It's organized as 128 sectors that are further divided into 1024 pages each (131,072 total pages). For compatibility with Flash memory devices, P5Q serial PCM supports SECTOR ERASE (128Kb sector) and BULK ERASE instructions.

In addition to BULK ERASE instructions, P5Q serial PCM supports four high-performance dual and quad input/output instructions that double or quadruple the transfer bandwidth for READ and PROGRAM operations.

- **DUAL OUTPUT FAST READ (DOFR)** instructions read data up to 66 MHz using both DQ0 and DQ1 pins as outputs.
- **QUAD OUTPUT FAST READ (QOFR)** instructions read data up to 50 MHz using DQ0, DQ1, DQ2(W#), and DQ3(HOLD#) pins as outputs.
- **DUAL INPUT FAST PROGRAM (DIFP)** instructions program data up to 66 MHz using both DQ0 and DQ1 pins as inputs.
- **QUAD INPUT FAST PROGRAM (QIFP)** instructions program data up to 50 MHz using DQ0, DQ1, DQ2(W#), and DQ3(HOLD#) pins as inputs.

PCM P5Q serial PCM can be write protected by software using a mix of volatile and nonvolatile protection features, depending on application needs. The protection granularity is 128Kb (sector granularity).

**Figure 1: Logic Diagram**

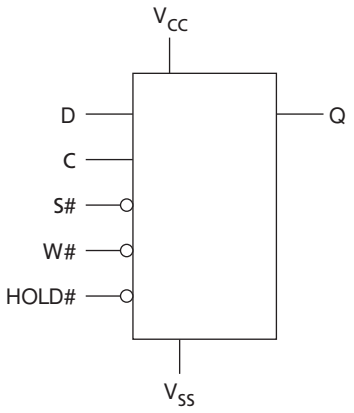
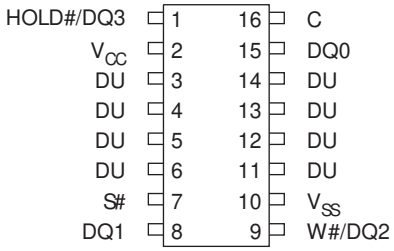



**Figure 2: SO16 Connections**




Notes:
1. DU = Do not use. User must float these pins.
2. See "Package Dimensions" on page 40 for package dimensions and how to identify pins.
3. For SO8 package solutions, contact your Micron representative.

# Signal Names

**Table 1: Signal Names**

| Signal Name | Standard x1 Mode | | Dual Mode | | Quad Mode | |
|---|---|---|---|---|---|---|
| | Function | Direction | Function | Direction | Function | Direction |
| C | Serial clock | Input | Serial clock | Input | Serial clock | Input |
| D (DQ0) | Serial data input | Input | Serial data I/O | I/O[1] | Serial data I/O | I/O[1] |
| Q (DQ1) | Serial data output | Output | Serial data I/O | I/O[1] | Serial data I/O | I/O[1] |
| S# | Chip select | Input | Chip select | Input | Chip select | Input |
| W# (DQ2) | Write protect | Input | Write Protect | Input | Serial data I/O | I/O[1] |
| HOLD# (DQ3) | Hold | Input | Hold | Input | Serial data I/O | I/O[1] |
| $V_{CC}$ | Supply voltage | | | | | |
| $V_{SS}$ | Ground | | | | | |

Notes: 1. Serves as an input during DUAL INPUT FAST PROGRAM (DIFP) and QUAD INPUT FAST PROGRAM (QIFP) instructions. Serves as an output during DUAL OUTPUT FAST READ (DOFR) and QUAD OUTPUT FAST READ (QOFR) instructions.

# Signal Descriptions

## Serial Data Input (D/DQ0)

The serial data input signal (D/DQ0) transfers data serially into the device and receives instructions, addresses, and the data to be programmed. Values are latched on the rising edge of serial clock (C).

During the DUAL OUTPUT FAST READ (DOFR) and QUAD OUTPUT FAST READ (QOFR) instructions, this pin is an output (DQ0). Data is shifted out on the falling edge of the C.

## Serial Data Output (Q/DQ1)

The serial data output signal (Q/DQ1) transfers data serially out of the device. Data is shifted out on the falling edge of C.

During the DIFP and QIFP instructions, this pin is used for data input (DQ1). It is latched on the rising edge of the C.

During the DOFR and QOFR instructions, this pin is used as data output (DQ1). Data is shifted out on the falling edge of C.

## Serial Clock (C)

The serial clock input signal (C) provides the timing of the serial interface. Instructions, addresses, or data present at DQ0 are latched on the rising edge of C. Data on DQ1 changes after the falling edge of C.

### Chip Select (S#)

When a chip select signal (S#) is HIGH, the device is deselected and DQ1 is High-Z. Unless an internal PROGRAM, ERASE, or WRITE STATUS REGISTER cycle is in progress, the device will be in standby power mode. Driving S# LOW enables the device, placing it in active power mode.

After power-up, a falling edge on S# is required prior to the start of any instruction.

### Hold (HOLD#/DQ3)

**The** hold signal (HOLD#) pauses any serial communications with the device without deselecting the device. During the HOLD condition, DQ1 is High-Z, and DQ0 and C are "Don't Care." To start the hold condition, the device must be selected with S# driven LOW.

During QIFP instructions, this pin is used for data input (DQ3). It is latched on the rising edge of the C. During QOFR instructions, this pin is used for data output (DQ3). Data is shifted out on the falling edge of C.

### Write Protect (W#/DQ2)

The write protect input signal (W#,DQ#2) freezes the size of the area of memory that is protected against program or erase instructions (as specified by the values in the BP3, BP2, BP1, and BP0 bits of the status register).

During QIFP instructions, this pin is used for data input (DQ2). It is latched on the rising edge of the C. During QOFR instructions, this pin is used for data output (DQ2). Data is shifted out on the falling edge of C.

### $V_{CC}$ Supply Voltage

$V_{CC}$ is the supply voltage.

### $V_{SS}$ Ground

$V_{SS}$ is the reference for the $V_{CC}$ supply voltage.

## SPI Modes

P5Q serial PCM devices can be driven by a microcontroller with its SPI peripheral running in either of these two modes:

- CPOL = 0, CPHA = 0
- CPOL = 1, CPHA = 1

For these two modes, input data is latched in on the rising edge of C, and output data is available from the falling edge of C. The difference between the two modes, as shown in Figure 4 on page 11, is the clock polarity when the bus master is in standby mode and not transferring data.

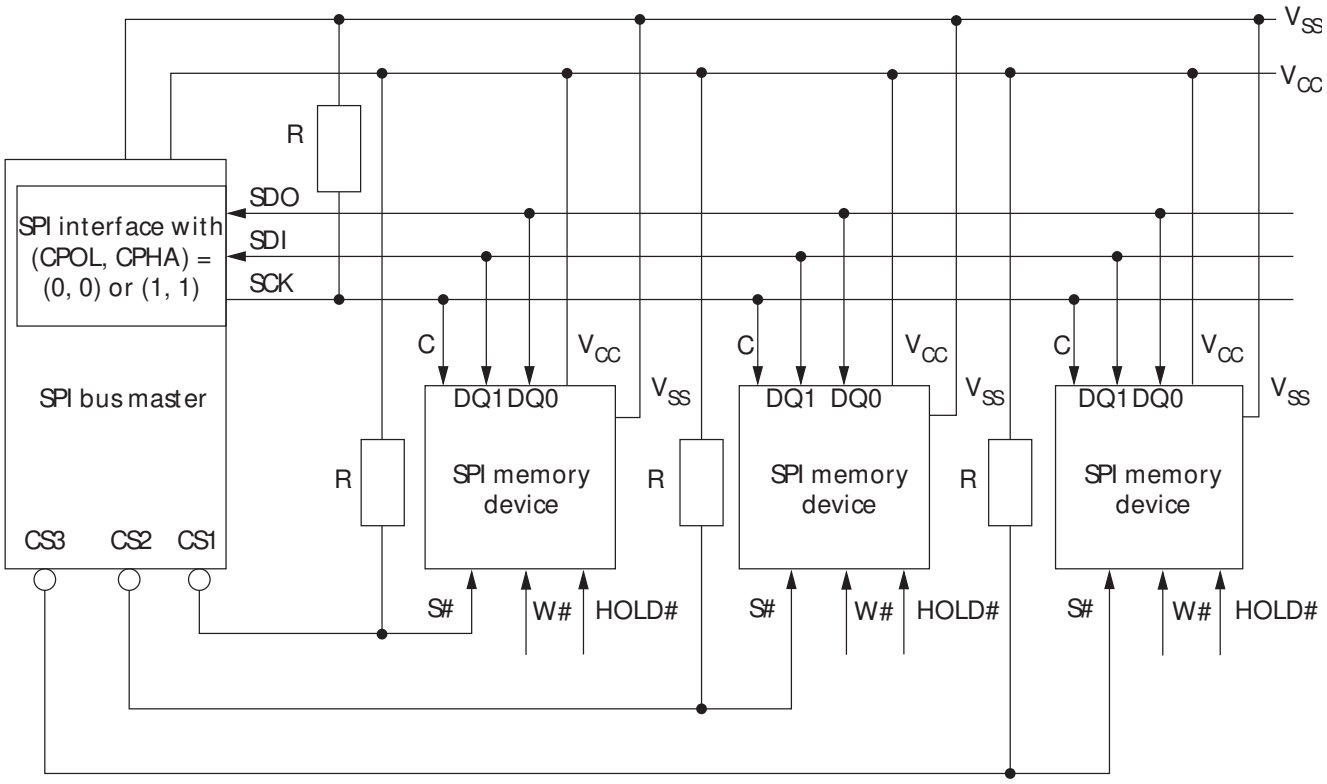
- C remains at 0 for (CPOL = 0, CPHA = 0)
- C remains at 1 for (CPOL = 1, CPHA = 1)

Figure 3 on page 10 is an example of three devices connected to an MCU on an SPI bus. Only one device is selected at a time, so only one device drives the serial data output (DQ1) line at a time; the other devices are High-Z. Resistors R (shown in Figure 3 on page 10) ensure that the P5Q serial PCM is not selected if the bus master leaves the S#

line in the High-Z state. Because the bus master may enter a state where all inputs/outputs are in High-Z at the same time (for example, when the bus master is reset), the clock line (C) must be connected to an external pull-down resistor. As a result, when all inputs/outputs become High-Z, the S# line is pulled HIGH, while the C line is pulled LOW. This ensures that S# and C do not become HIGH at the same time and that the $^{t}$SHCH requirement is met.

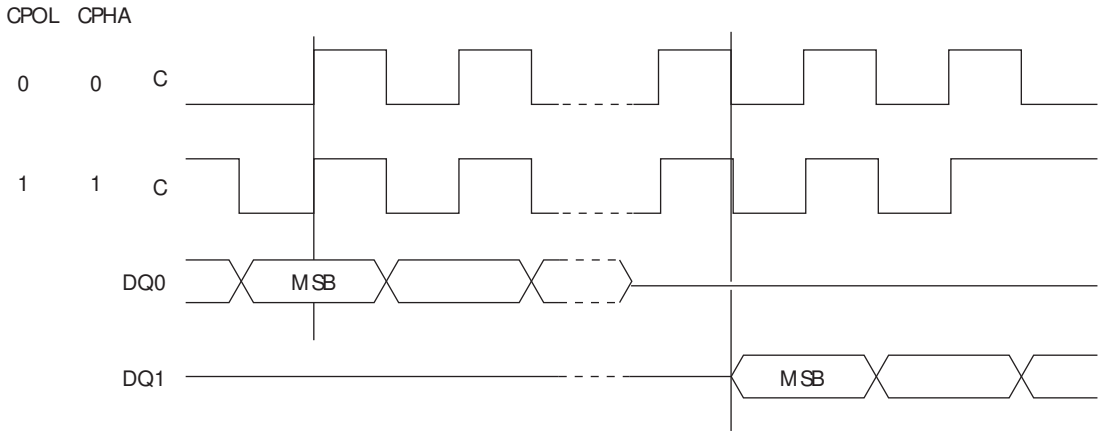
The typical value of R is 100kΩ, assuming that the time constant $R \times C_p$ ($C_p$ = parasitic capacitance of the bus line) is shorter than the time during which the bus master leaves the SPI bus in High-Z.

**Figure 3: Bus Master and Memory Devices on the SPI Bus**




Notes: 1. W# and (HOLD# signals should be driven HIGH or LOW, as appropriate.

**Figure 4: SPI Modes Supported**



# Operating Features

To better understand operating features of the P5Q serial PCM device, refer to the following definitions:

- **PROGRAM**: P5Q serial PCM devices write only 0s of the user data to the array and treat 1s as data masks. This is similar to programming on a floating gate Flash device.
- **Bit-alterable WRITE**: P5Q serial PCM devices write both 0s and 1s of the user data to the array.
- PROGRAM on all 1s: Only 0s are written to the array, and 1s are treated as data masks. PROGRAM on all 1s also requires that the entire page being written be previously set to all 1s. PROGRAM on all 1s is also referred to as PRESET WRITE.

## PAGE PROGRAM

To PROGRAM/WRITE one data byte, two instructions are required: WRITE ENABLE (WREN), which is one byte; and a PAGE PROGRAM (PP) sequence, which consists of four bytes plus data byte. This is followed by the internal PROGRAM cycle (of duration $^{t}$PP).

To spread this overhead, the PP instruction allows up to 64 bytes to be programmed/written at a time, provided that they lie in consecutive addresses on the same page of memory.

For optimized timings, it is recommended to use the PP instruction to program all consecutive targeted bytes in a single sequence versus using several PP sequences with each containing only a few bytes (see "PAGE PROGRAM (PP)" on page 27 and Table 16 on page 37).

## DUAL INPUT FAST PROGRAM

The DUAL INPUT FAST PROGRAM (DIFP) instruction makes it possible to PROGRAM/WRITE up to 64 bytes using two input pins at the same time.

For optimized timings, it is recommended to use the DIFP instruction to program all consecutive targeted bytes in a single sequence rather than using several DIFP sequences each containing only a few bytes.

## QUAD INPUT FAST PROGRAM

The QUAD INPUT FAST PROGRAM (QIFP) instruction makes it possible to PROGRAM/WRITE up to 64 bytes using four input pins at the same time.

For optimized timings, use the QIFP instruction to program all consecutive targeted bytes in a single sequence rather than several QIFP sequences each containing only a few bytes.

## SECTOR ERASE and BULK ERASE

A sector can be erased to all 1s (FFh) at a time using the SECTOR ERASE (SE) instruction. The entire memory can be erased using the BULK ERASE (BE) instruction. This starts an internal ERASE cycle (of duration $^{t}$SE or $^{t}$BE).

The ERASE instruction must be preceded by a WREN instruction.

## Polling During a WRITE, PROGRAM, or ERASE Cycle

Additional improvements in the time to WRSR, PP, DIFP, QIFP, or ERASE (SE or BE) can be achieved by not waiting for the worst case delay ($^{t}$W, $^{t}$PP, $^{t}$SMEN, $^{t}$SMEX, $^{t}$SE, or $^{t}$BE). The write in progress (WIP) bit is provided in the status register so that the application program can monitor its value, polling it to establish when the previous WRITE cycle, PROGRAM cycle, or ERASE cycle is complete.

## Active Power and Standby Power

When S# is LOW, the device is selected and is in the active power mode. When S# is HIGH, the device is deselected, but could remain in the active power mode until all internal cycles have completed (PROGRAM, ERASE, WRITE STATUS REGISTER). The device then goes in to the standby power mode. The device consumption drops to $I_{CC1}$.

## Status Register

The status register contains a number of status and control bits that can be read or set (as appropriate) by specific instructions. See “READ STATUS REGISTER (RDSR)” on page 20 for a detailed description of the status register bits.

## Protocol-Related Protections

The environments where nonvolatile memory devices are used can be very noisy, but SPI devices cannot operate correctly in the presence of excessive noise. To help combat this, the P5Q serial PCM features the following data protection mechanisms:

- Power on reset and an internal timer ($^{t}$PUW) can provide protection against inadvertent changes while the power supply is outside the operating specification.
- PROGRAM, ERASE, and WRITE STATUS REGISTER are checked to ensure they consist of a number of clock pulses that is a multiple of eight before they are accepted for execution.
- All instructions that modify data must be preceded by a WREN instruction to set the WEL bit. This bit is returned to its reset state by the following events:
  - Power-up
  - WRDI instruction completion
  - WRSR instruction completion
  - PP instruction completion
  - DIFP instruction completion
  - QIFP instruction completion
  - SE instruction completion
  - BE instruction completion
- The block protect bits and top/bottom bit enable part of the memory to be configured as read-only. This is the software protect mode (SPM).
- The W# signal enables the block protect bits (BP3, BP2, BP1, BP0), top/bottom (TB) bit, and status register write disable (SRWD) bit to be protected. This is the hardware protected mode (HPM).

**Table 2: Sizes of Protected Areas**

| Status Register Contents | | | | | Memory Content | |
|---|---|---|---|---|---|---|
| TB Bit | BP Bit 3 | BP Bit 2 | BP Bit 1 | BP Bit 0 | Protected Area | Unprotected Area |
| 0 | 0 | 0 | 0 | 0 | None | All sectors[1] (sectors 0 to 127) |
| 0 | 0 | 0 | 0 | 1 | Upper 128 (sector 127) | Sectors 0 to 126 |
| 0 | 0 | 0 | 1 | 0 | Upper 64 (sectors 126 to 127) | Sectors 0 to 125 |
| 0 | 0 | 0 | 1 | 1 | Upper 32 (sectors 124 to 127) | Sectors 0 to 123 |
| 0 | 0 | 1 | 0 | 0 | Upper 16 (sectors 120 to 127) | Sectors 0 to 119 |
| 0 | 0 | 1 | 0 | 1 | Upper 8 (sectors 112 to 127) | Sectors 0 to 111 |
| 0 | 0 | 1 | 1 | 0 | Upper quarter (sectors 96 to 127) | Sectors 0 to 95 |
| 0 | 0 | 1 | 1 | 1 | Upper half (sectors 64 to 127) | Sectors 0 to 63 |
| 0 | 1 | X[2] | X[2] | X[2] | All sectors (sectors 0 to 127) | None |
| 1 | 0 | 0 | 0 | 0 | None | All sectors[1] (sectors 0 to 127) |
| 1 | 0 | 0 | 0 | 1 | Lower 128 (sector 0) | Sectors 1 to 127 |
| 1 | 0 | 0 | 1 | 0 | Lower 64 (sectors 0 to 1) | Sectors 2 to 127 |
| 1 | 0 | 0 | 1 | 1 | Lower 32 (sectors 0 to 3) | Sectors 4 to 127 |
| 1 | 0 | 1 | 0 | 0 | Lower 16 (sectors 0 to 7) | Sectors 8 to 127 |
| 1 | 0 | 1 | 0 | 1 | Lower 8 (sectors 0 to 15) | Sectors 16 to 127 |
| 1 | 0 | 1 | 1 | 0 | Lower 4 (sectors 0 to 31) | Sectors 32 to 127 |
| 1 | 0 | 1 | 1 | 1 | Lower half (sectors 0 to 63) | Sectors 64 to 127 |
| 1 | 1 | X[2] | X[2] | X[2] | All sectors (sectors 0 to 127) | None |

Notes:
1. The device is ready to accept a BULK ERASE instruction if all block protect bits (BP3, BP2, BP1, BP0) are 0.
2. X can be 0 or 1.

## Hold Condition

The Hold (HOLD#) signal is used to pause any serial communications with the device without resetting the clocking sequence. However, taking this signal LOW does not terminate any WRITE STATUS REGISTER, PROGRAM, or ERASE cycle that is currently in progress.

To enter the hold condition, the device must be selected, with S# LOW. The hold condition starts on the falling edge of the HOLD# signal, provided that this coincides with C being LOW (as shown in Figure 5 on page 15). The hold condition ends on the rising edge of the HOLD# signal, provided that this coincides with C being LOW.

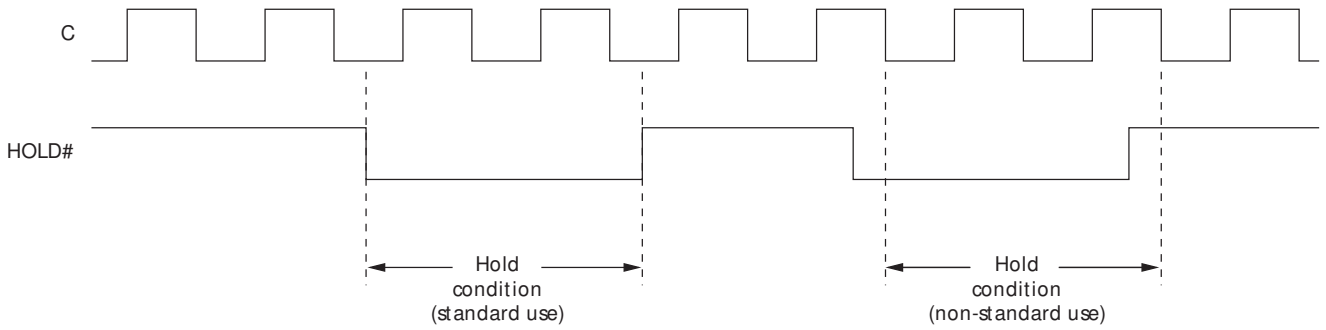
If the falling edge does not coincide with C being LOW, the hold condition starts after the next time C goes LOW. Similarly, if the rising edge does not coincide with C being LOW, the hold condition ends after C next goes LOW (as shown in Figure 5 on page 15).

During the hold condition, DQ1 is High-Z, and DQ0 and C are "Don't Care."

Normally, the device is kept selected, with S# driven LOW, for the whole duration of the hold condition. This is to ensure that the state of the internal logic remains unchanged from the moment of entering the hold condition.

If S# goes HIGH while the device is in the hold condition, this has the effect of resetting the internal logic of the device. To restart communication with the device, it is necessary to drive HOLD# HIGH, and then to drive S# LOW. This prevents the device from going back to the hold condition.

**Figure 5: Hold Condition Activation**




# Memory Organization

The memory is organized as:

- 16,772,216 bytes (8 bits each)
- 8 super page programming regions (16 sectors each)
- 128 sectors (128Kb each)
- 262,144 pages (64 bytes each)

Each page can be individually programmed (bits are programmed from 1 to 0) or written (bit alterable: 1 can be altered to 0 and 0 can be altered to 1). The device is sector or bulk erasable (bits are erased from 0 to 1).

**Table 3: Memory Organization**

| Sector | Address Range | |
|---|---|---|
| 127 | FE0000 | FFFFFF |
| 126 | FC0000 | FDFFFF |
| 125 | FA0000 | FBFFFF |
| 124 | F80000 | F9FFFF |
| 123 | F60000 | F7FFFF |
| 122 | F40000 | F5FFFF |
| 121 | F20000 | F3FFFF |
| 120 | F00000 | F1FFFF |
| 119 | EE0000 | EFFFFF |
| 118 | EC0000 | EDFFFF |
| 117 | EA0000 | EBFFFF |
| 116 | E80000 | E9FFFF |
| 115 | E60000 | E7FFFF |
| 114 | E40000 | E5FFFF |
| 113 | E20000 | E3FFFF |
| 112 | E00000 | E1FFFF |
| 111 | DE0000 | DFFFFF |
| 110 | DC0000 | DDFFFF |
| 109 | DA0000 | DBFFFF |
| 108 | D80000 | D9FFFF |
| 107 | D60000 | D7FFFF |
| 106 | D40000 | D5FFFF |
| 105 | D20000 | D3FFFF |

| Sector | Address Range | |
|---|---|---|
| 102 | CC0000 | CDFFFF |
| 101 | CA0000 | CBFFFF |
| 100 | C80000 | C9FFFF |
| 99 | C60000 | C7FFFF |
| 98 | C40000 | C5FFFF |
| 97 | C20000 | C3FFFF |
| 96 | C00000 | C1FFFF |
| 95 | BE0000 | BFFFFF |
| 94 | BC0000 | BDFFFF |
| 93 | BA0000 | BBFFFF |
| 92 | B80000 | B9FFFF |
| 91 | B60000 | B7FFFF |
| 90 | B40000 | B5FFFF |
| 89 | B20000 | B3FFFF |
| 88 | B00000 | B1FFFF |
| 87 | AE0000 | AFFFFF |
| 86 | AC0000 | ADFFFF |
| 85 | AA0000 | ABFFFF |
| 84 | A80000 | A9FFFF |
| 83 | A60000 | A7FFFF |
| 82 | A40000 | A5FFFF |
| 81 | A20000 | A3FFFF |
| 80 | A00000 | A1FFFF |

**Table 3: Memory Organization (continued)**

| Sector | Address Range | |
|---|---|---|
| 104 | D00000 | D1FFFF |
| 103 | CE0000 | CFFFFF |
| 77 | 9A0000 | 9BFFFF |
| 76 | 980000 | 99FFFF |
| 75 | 960000 | 97FFFF |
| 74 | 940000 | 95FFFF |
| 73 | 920000 | 93FFFF |
| 72 | 900000 | 91FFFF |
| 71 | 8E0000 | 8FFFFF |
| 70 | 8C0000 | 8DFFFF |
| 69 | 8A0000 | 8BFFFF |
| 68 | 880000 | 89FFFF |
| 67 | 860000 | 87FFFF |
| 66 | 840000 | 85FFFF |
| 65 | 820000 | 83FFFF |
| 64 | 800000 | 81FFFF |
| 63 | 7E0000 | 7FFFFF |
| 62 | 7C0000 | 7DFFFF |
| 61 | 7A0000 | 7BFFFF |
| 60 | 780000 | 79FFFF |
| 59 | 760000 | 77FFFF |
| 58 | 740000 | 75FFFF |
| 57 | 720000 | 73FFFF |
| 56 | 700000 | 71FFFF |
| 55 | 6E0000 | 6FFFFF |
| 54 | 6C0000 | 6DFFFF |
| 53 | 6A0000 | 6BFFFF |
| 52 | 680000 | 69FFFF |
| 51 | 660000 | 67FFFF |
| 50 | 640000 | 65FFFF |
| 49 | 620000 | 63FFFF |
| 48 | 600000 | 61FFFF |
| 47 | 5E0000 | 5FFFFF |
| 46 | 5C0000 | 5DFFFF |
| 45 | 5A0000 | 5BFFFF |
| 44 | 580000 | 59FFFF |
| 43 | 560000 | 57FFFF |
| 7 | 0E0000 | 0FFFFF |
| 6 | 0C0000 | 0DFFFF |
| 5 | 0A0000 | 0BFFFF |
| 4 | 080000 | 09FFFF |

| Sector | Address Range | |
|---|---|---|
| 79 | 9E0000 | 9FFFFF |
| 78 | 9C0000 | 9DFFFF |
| 42 | 540000 | 55FFFF |
| 41 | 520000 | 53FFFF |
| 40 | 500000 | 51FFFF |
| 39 | 4E0000 | 4FFFFF |
| 38 | 4C0000 | 4DFFFF |
| 37 | 4A0000 | 4BFFFF |
| 36 | 480000 | 49FFFF |
| 35 | 460000 | 47FFFF |
| 34 | 440000 | 45FFFF |
| 33 | 420000 | 43FFFF |
| 32 | 400000 | 41FFFF |
| 31 | 3E0000 | 3FFFFF |
| 30 | 3C0000 | 3DFFFF |
| 29 | 3A0000 | 3BFFFF |
| 28 | 380000 | 39FFFF |
| 27 | 360000 | 37FFFF |
| 26 | 340000 | 35FFFF |
| 25 | 320000 | 33FFFF |
| 24 | 300000 | 31FFFF |
| 23 | 2E0000 | 2FFFFF |
| 22 | 2C0000 | 2DFFFF |
| 21 | 2A0000 | 2BFFFF |
| 20 | 280000 | 29FFFF |
| 19 | 260000 | 27FFFF |
| 18 | 240000 | 25FFFF |
| 17 | 220000 | 23FFFF |
| 16 | 200000 | 21FFFF |
| 15 | 1E0000 | 1FFFFF |
| 14 | 1C0000 | 1DFFFF |
| 13 | 1A0000 | 1BFFFF |
| 12 | 180000 | 19FFFF |
| 11 | 160000 | 17FFFF |
| 10 | 140000 | 15FFFF |
| 9 | 120000 | 13FFFF |
| 8 | 100000 | 11FFFF |
| 3 | 060000 | 07FFFF |
| 2 | 040000 | 05FFFF |
| 1 | 020000 | 03FFFF |
| 0 | 000000 | 01FFFF |

**Table 4: Organization of Super Page Regions**

| Programming Region | Sectors | Address Range |
|---|---|---|
| 7 | 112 to 127 | E00000 to FFFFFF |
| 6 | 96 to 111 | C00000 to DFFFFF |
| 5 | 80 to 95 | A00000 to BFFFFF |
| 4 | 64 to 79 | 800000 to 9FFFFF |
| 3 | 48 to 63 | 600000 to 7FFFFF |
| 2 | 32 to 47 | 400000 to 5FFFFF |
| 1 | 16 to 31 | 200000 to 3FFFFF |
| 0 | 0 to 15 | 000000 to 1FFFFF |

# Instructions

All instructions, addresses, and data are shifted in and out of the device, most significant bit first.

Serial data input DQ0 is sampled on the first rising edge of C after S# is driven LOW. Then, the one-byte instruction code must be shifted in to the device, most significant bit first, on serial data input DQ0, each bit being latched on the rising edges of C. The instruction set is listed in Table 5 on page 17.

Every instruction sequence starts with a one-byte instruction code. Depending on the instruction, this might be followed by address bytes, data bytes, both, or none.

In the case of read data bytes (READ), read data bytes at higher speed (FAST_READ), DOFR, QOFR, RDSR, or READ IDENTIFICATION (RDID) instruction, the shifted-in instruction sequence is followed by a data-out sequence. S# can be driven HIGH after any bit of the data-out sequence is being shifted out.

In the case of a PP, DIFP, QIFP, SE, BE, WRSR, WREN, or WRDI, S# must be driven HIGH exactly at a byte boundary; otherwise the instruction is rejected and is not executed. That is, S# must be driven HIGH when the number of clock pulses after S# being driven LOW is an exact multiple of eight.

All attempts to access the memory array during a WRITE STATUS REGISTER cycle, PROGRAM cycle, or ERASE cycle are ignored and the internal WRITE STATUS REGISTER cycle, PROGRAM cycle, ERASE cycle continues unaffected.

**Note:** Output High-Z is defined as the point where data out is no longer driven.

**Table 5: Instruction Set**

| Instruction | Description | One-Byte Instruction Code | | Address Bytes | Dummy Bytes | Data Bytes |
|---|---|---|---|---|---|---|
| WREN | Write enable | 0000 0110 | 06h | 0 | 0 | 0 |
| WRDI | Write disable | 0000 0100 | 04h | 0 | 0 | 0 |
| RDID | Read identification | 1001 1111 | 9Fh | 0 | 0 | 1 to 3 |
| | | 1001 1110 | 9Eh | 0 | 0 | 1 to 3 |
| RDSR | Read status register | 0000 0101 | 05h | 0 | 0 | 1 to ∞ |
| WRSR | Write status register | 0000 0001 | 01h | 0 | 0 | 1 |
| READ | Read data bytes | 0000 0011 | 03h | 3 | 0 | 1 to ∞ |
| FAST_READ | Read data bytes at higher speed | 0000 1011 | 0Bh | 3 | 1 | 1 to ∞ |
| DOFR | Dual output fast read | 0011 1011 | 3Bh | 3 | 1 | 1 to ∞ |

**Table 5: Instruction Set (continued)**

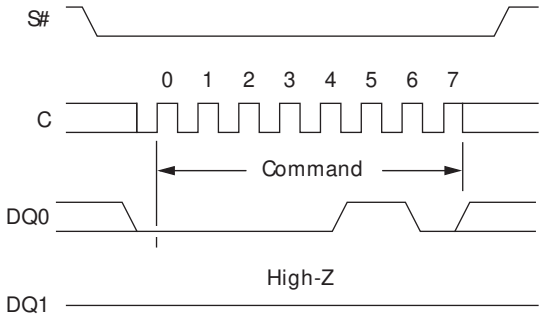
| Instruction | Description | One-Byte Instruction Code | | Address Bytes | Dummy Bytes | Data Bytes |
|---|---|---|---|---|---|---|
| QOFR | Quad output fast read | 0110 1011 | 6Bh | 3 | 1 | 1 to ∞ |
| PP | Page program (legacy program) | 0000 0010 | 02h | 3 | 0 | 1 to 64 |
| | Page program (bit-alterable write) | 0010 0010 | 22h | 3 | 0 | 1 to 64 |
| | Page program (on all 1s) | 1101 0001 | D1h | 3 | 0 | 1 to 64 |
| DIFP | Dual input fast program (legacy program) | 1010 0010 | A2h | 3 | 0 | 1 to 64 |
| | Dual input fast program (bit-alterable write) | 1101 0011 | D3h | 3 | 0 | 1 to 64 |
| | Dual input fast program (on all 1s) | 1101 0101 | D5h | 3 | 0 | 1 to 64 |
| QIFP | Quad input fast program (legacy program) | 0011 0010 | 32h | 3 | 0 | 1 to 64 |
| | Quad input fast program (bit-alterable write) | 1101 0111 | D7h | 3 | 0 | 1 to 64 |
| | Quad input fast program (on all 1s) | 1101 1001 | D9h | 3 | 0 | 1 to 64 |
| SE | Sector erase | 1101 1000 | D8h | 3 | 0 | 0 |
| BE | Bulk erase | 1100 0111 | C7h | 0 | 0 | 0 |

## WRITE ENABLE (WREN)

The WRITE ENABLE (WREN) instruction sets the WEL bit. The WEL bit must be set prior to every PP, DIFP, SE, BE, or WRSR instruction.

The WREN instruction is entered by driving S# LOW, sending the instruction code, and then driving S# HIGH.

**Figure 6: WRITE ENABLE (WREN) Instruction Sequence**




## WRITE DISABLE (WRDI)

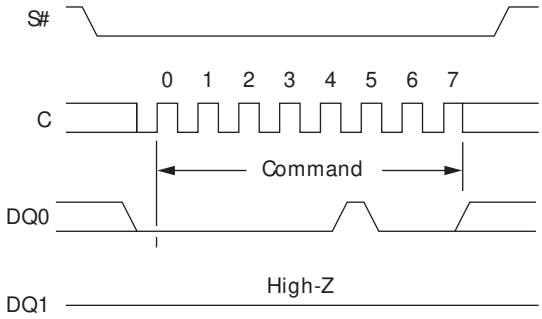
The WRITE DISABLE (WRDI) instruction resets the WEL bit. The WRDI instruction is entered by driving S# LOW, sending the instruction code, and then driving S# HIGH.

The WEL bit is reset under the following conditions:

- Power-up
- WRDI instruction completion
- WRSR instruction completion
- PP instruction completion
- DIFP instruction completion

- QIFP instruction completion
- SE instruction completion
- BE instruction completion

**Figure 7: WRITE DISABLE (WRDI) Instruction Sequence**




## READ IDENTIFICATION (RDID)

The READ IDENTIFICATION (RDID) instruction enables devices to read the device identification data, including manufacturer identification (1 byte) and device identification (2 bytes).

The manufacturer identification is assigned by JEDEC and has the value 20h for Micron. Any RDID instruction while an ERASE or PROGRAM cycle is in progress is not decoded and has no effect on the cycle that is in progress.

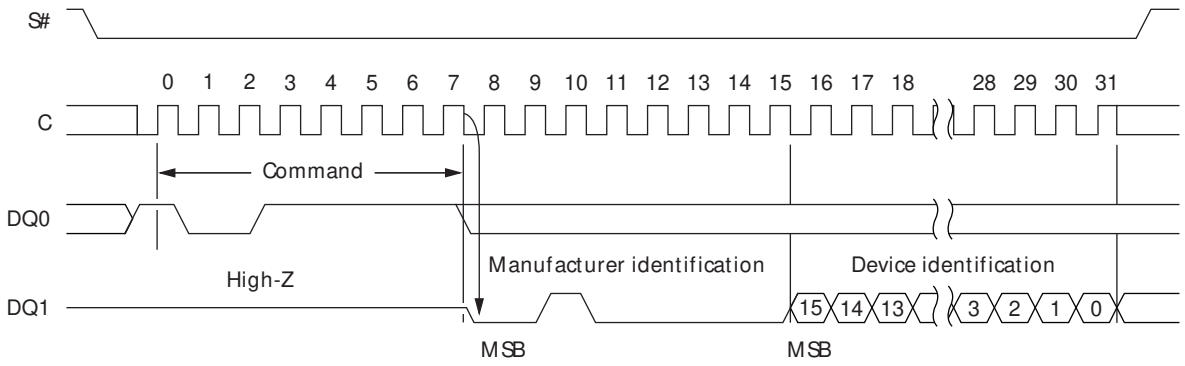
The device is first selected by driving S# LOW. Then, the 8-bit instruction code for the instruction is shifted in. After this, the 24-bit device identification stored in the memory will be shifted out on serial data output (DQ1). Each bit is shifted out during the falling edge of C. The instruction sequence is shown in Figure 8 on page 20.

The RDID instruction is terminated by driving S# HIGH at any time during data output. When S# is driven HIGH, the device is put in the standby power mode. After the device is in the standby power mode, the device waits to be selected so that it can receive, decode, and execute instructions.

**Table 6: READ IDENTIFICATION (RDID) Data-Out Sequence**

| Manufacturer Identification | Device identification | |
|---|---|---|
| | Memory Type (Upper Byte) | Memory Capacity (Lower Byte) |
| 20h | DAh | 18h |

**Figure 8: READ IDENTIFICATION (RDID) Instruction Sequence and Data-Out Sequence**

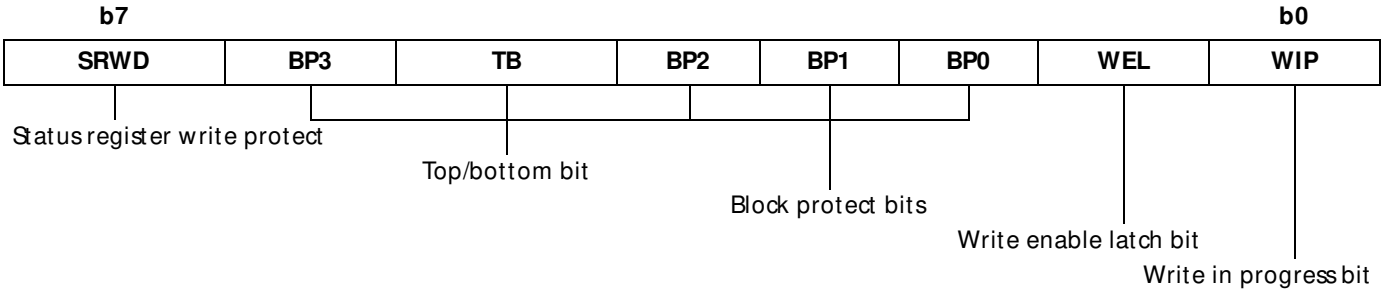



## READ STATUS REGISTER (RDSR)

The READ STATUS REGISTER (RDSR) instruction enables the status register to be read. The status register may be read at any time, even while a PROGRAM, ERASE, or WRITE STATUS REGISTER cycle is in progress. When one of these cycles is in progress, it is recommended to check the WIP bit before sending a new instruction to the device. It is also possible to read the status register continuously, as shown in Figure 9 on page 21.

RDSR is the only instruction accepted by the device while a PROGRAM, ERASE, WRITE STATUS REGISTER operation is in progress.

**Table 7: Status Register Format**




The status and control bits of the status register are described below.

### WIP Bit

The write in progress (WIP) bit indicates whether the memory is busy with a WRITE STATUS REGISTER, PROGRAM, or ERASE cycle. When set to 1, one of these cycles is in progress; when reset to 0, none of these cycles is in progress. While WIP is 1, RDSR is the only instruction the device will accept; all other instructions are ignored.

### WEL Bit

The write enable latch (WEL) bit indicates the status of the internal write enable latch. When set to 1, the internal write enable latch is set. When it is set to 0, the internal write enable latch is reset, and no WRITE STATUS REGISTER, PROGRAM, or ERASE instruction is accepted.

### Block Protect Bits

The block protect bits (BP3, BP2, BP1, BP0) are nonvolatile. They define the size of the area to be software protected against PROGRAM (or WRITE) and ERASE instructions. These bits are written with the WRSR instruction. When one or more of the block protect bits is set to 1, the relevant memory area (as defined in Table 2 on page 14) becomes protected against PP, DIFP, QIFP, and SE instructions. The block protect bits can be written, provided that the hardware protected mode has not been set. The BE instruction is executed if all block protect bits are 0.

### Top/Bottom Bit

The top/bottom (TB) bit is nonvolatile. It can be set and reset with the WRSR instruction, provided that the WREN instruction has been issued. The TB bit is used in conjunction with the block protect bits to determine if the protected area defined by the block protect bits starts from the top or the bottom of the memory array.

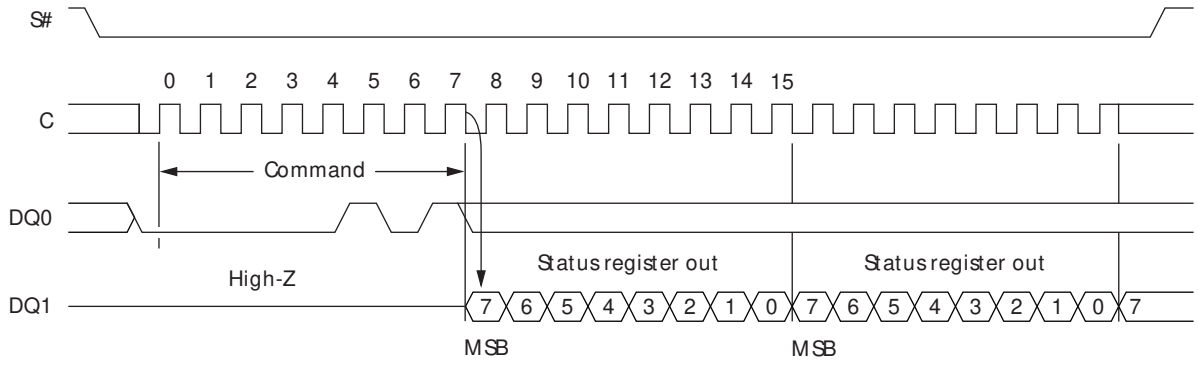
- When TB bit is reset to 0 (default value), the area protected by the block protect bits starts from the top of the memory array (see Table 2 on page 14)
- When TB bit is set to 1, the area protected by the block protect bits starts from the bottom of the memory array (see Table 2 on page 14).

The TB bit cannot be written when the SRWD bit is set to 1 and the W# pin is driven LOW.

### SRWD Bit

The status register write disable (SRWD) bit is operated in conjunction with the W# signal. The SRWD bit and the W# signal allow the device to be put in the hardware protected mode (when the SRWD bit is set to 1 and W# is driven LOW). In this mode, the nonvolatile bits of the status register (SRWD, TB, BP3, BP2, BP1, BP0) become read-only bits and the WRSR instruction is no longer accepted for execution.

**Figure 9: READ STATUS REGISTER (RDSR) Instruction Sequence and Data-Out Sequence**




## WRITE STATUS REGISTER (WRSR)

The WRITE STATUS REGISTER (WRSR) instruction enables new values to be written to the status register. Before it can be accepted, a WREN instruction must previously have been executed. After the WREN instruction has been decoded and executed, the device sets the WEL.

The WRSR instruction is entered by driving S# LOW, followed by the instruction code and the data byte on serial data input (DQ0). The instruction sequence is shown in Figure 10 on page 22. The WRSR instruction has no effect on B1 and B0 of the status register.

S# must be driven HIGH after the eighth bit of the data byte has been latched in. If not, the WRSR instruction is not executed. As soon as S# is driven HIGH, the self-timed WRITE STATUS REGISTER cycle (whose duration is $^{t}W$) is initiated. While the WRITE STATUS REGISTER cycle is in progress, the status register may still be read to check the value of the WIP bit. The WIP bit is 1 during the self-timed WRITE STATUS REGISTER cycle, and is 0 when it is completed. When the cycle is completed, the WEL is reset.

The WRSR instruction enables the user to change the values of the block protect bits and to define the size of the area that is to be treated as read-only, as defined in Table 2 on page 14. The WRSR instruction also enables the user to set and reset the SRWD bit in accordance with the W# signal. The SRWD bit and W# signal enable the device to be put in the hardware protected mode (HPM). The WRSR instruction is not executed after the HPM is entered.

RDSR is the only instruction that is accepted while a WRSR operation is in progress; all other instructions are ignored.

**Figure 10: WRITE STATUS REGISTER (WRSR) Instruction Sequence**

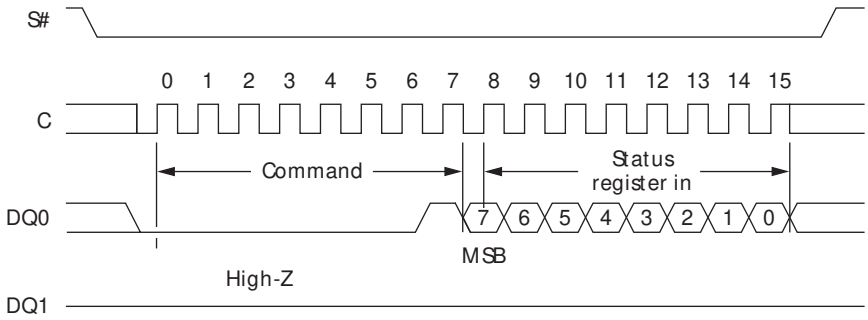


**Table 8: Protection Modes**

| W# | SRWD Bit | Mode | Write Protection of Status Register | Memory Content | |
|---|---|---|---|---|---|
| | | | | Protected Area[1] | Unprotected Area[1] |
| 1 | 0 | Software protected (SPM) | Status register is writable (if the WREN instruction has set the WEL bit); the values in the SRWD, TB, BP3, BP2, BP1, and BP0 bits can be changed | Protected against PAGE PROGRAM, SECTOR ERASE, and BULK ERASE | Ready to accept PAGE PROGRAM, and SECTOR ERASE instructions |
| 0 | 0 | | | | |
| 1 | 1 | | | | |
| 0 | 1 | Hardware protected (HPM) | Status register is hardware write protected; the values in the SRWD, TB, BP3, BP2, BP1, and BP0 bits cannot be changed | Protected against PAGE PROGRAM, SECTOR ERASE, and BULK ERASE | Ready to accept PAGE PROGRAM, and SECTOR ERASE instructions |

Notes: 1. As defined by the values in the block protect bits (BP3, BP2, BP1, BP0) of the status register, as shown in Table 2 on page 14.

When the SRWD bit of the status register is 0 (its initial delivery state), it is possible to write to the status register, provided that the WEL bit has previously been set by a WREN instruction, regardless of whether W# is driven HIGH or LOW.

When the SRWD bit of the status register is set to 1, two cases need to be considered, depending on the state of W#:

- If W# is driven HIGH, it is possible to write to the status register provided that the WEL bit has previously been set by a WREN instruction.
- If W# is driven LOW, it is not possible to write to the status register even if the WEL bit has previously been set by a WREN instruction (attempts to write to the status register are rejected and are not accepted for execution). As a consequence, all the data bytes in the memory area that are software protected (SPM) by the block protect bits of the status register are also hardware protected against data modification.

Regardless of the order of the two events, the HPM can be entered in one of two ways:

- Set the SRWD bit after driving W# LOW.
- Drive W# LOW after setting the SRWD bit.

The only way to exit HPM after it has been entered is to pull write protect (W#) HIGH.

If write protect (W#) is permanently tied HIGH, HPM can never be activated, and only the software protected mode (SPM), using the block protect bits of the status register, can be used.

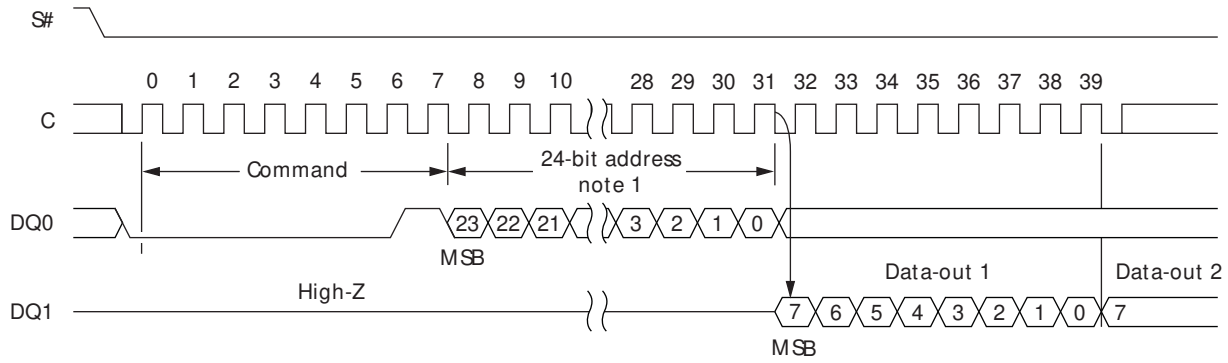
## Read Data Bytes (READ)

The device is first selected by driving S# LOW. The instruction code for the read data bytes (READ) instruction is followed by a 3-byte address A[23:0], with each bit being latched in during the rising edge of C. Then the memory contents at that address are shifted out on serial data output (DQ1), with each bit being shifted out at a maximum frequency $^{f}R$, during the falling edge of C. The instruction sequence is shown in Figure 11 on page 24.

The first byte addressed can be at any location. The address is automatically incremented to the next higher address after each byte of data is shifted out. The whole memory can be read with a single READ instruction. When the highest address is reached, the address counter rolls over to 000000h, enabling the read sequence to be continued indefinitely.

The READ instruction is terminated by driving S# HIGH. S# can be driven HIGH at any time during data output. Any READ instruction, while an ERASE, PROGRAM, or WRITE is in progress, is rejected without having any effects on the cycle that is in progress.

**Figure 11: READ DATA BYTES (READ) Instruction Sequence and Data-Out Sequence**



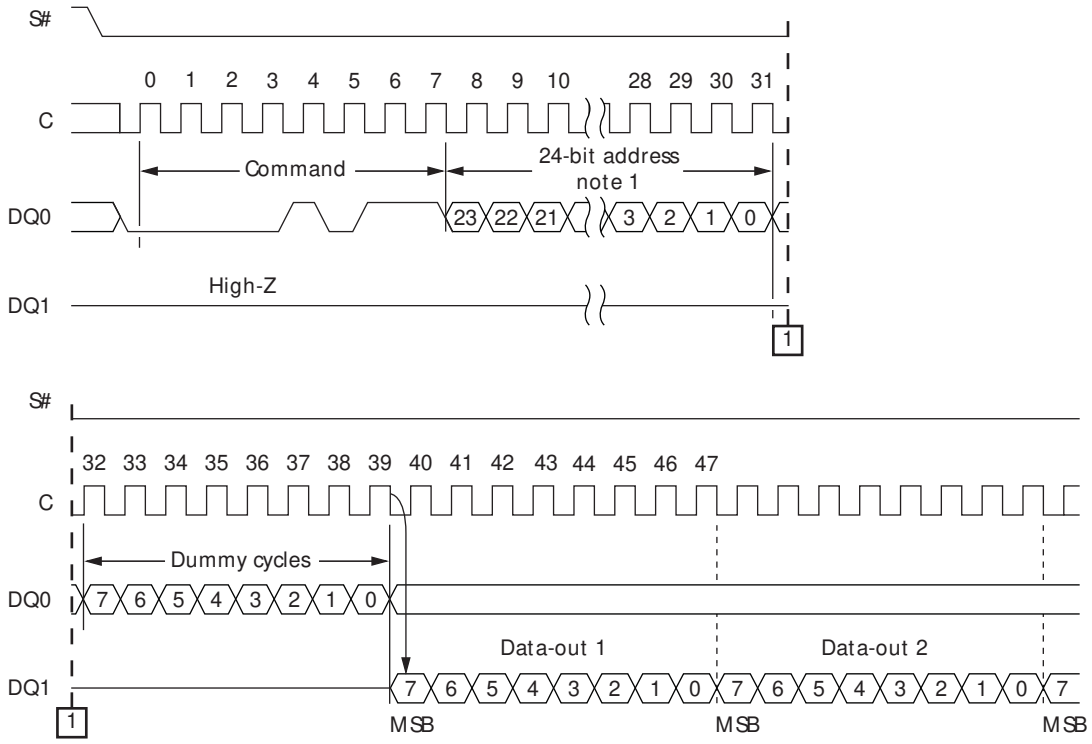

## Read Data Bytes at Higher Speed (FAST_READ)

The device is first selected by driving S# LOW. The instruction code for the read data bytes at higher speed (FAST_READ) instruction is followed by a 3-byte address A[23:0] and a dummy byte, with each bit being latched in during the rising edge of C. Then the memory contents at that address are shifted out on serial data output (DQ1) at a maximum frequency $^{f}C$, during the falling edge of C. The instruction sequence is shown in Figure 12 on page 25.

The first byte addressed can be at any location. The address is automatically incremented to the next higher address after each byte of data is shifted out. The whole memory can be read with a single FAST_READ instruction. When the highest address is reached, the address counter rolls over to 000000h, enabling the read sequence to be continued indefinitely.

The FAST_READ instruction is terminated by driving S# HIGH. S# can be driven HIGH at any time during data output. While an ERASE, PROGRAM, or WRITE cycle is in progress, any FAST_READ instruction is rejected without having any effects on the cycle that is in progress.

**Figure 12: FAST_READ Instruction Sequence and Data-Out Sequence**




## DUAL OUTPUT FAST READ (DOFR)

The DUAL OUTPUT FAST READ (DOFR) instruction is very similar to the FAST_READ instruction, except that the data are shifted out on two pins (DQ0 and DQ1) instead of one. Outputting the data on two pins instead of one doubles the data transfer bandwidth compared to the outputting data using the FAST_READ instruction.

The device is first selected by driving S# LOW. The instruction code for the DOFR instruction is followed by a 3-byte address A[23:0] and a dummy byte, with each bit being latched-in during the rising edge of serial clock (C). Then the memory contents at that address are shifted out on DQ0 and DQ1 at a maximum frequency $^{f}C$, during the falling edge of C. The instruction sequence is shown in Figure 13 on page 26.

The first byte addressed can be at any location. The address is automatically incremented to the next higher address after each byte of data is shifted out on DQ0 and DQ1. The whole memory can be read with a single DOFR instruction. When the highest address is reached, the address counter rolls over to 00 0000h so that the read sequence can be continued indefinitely.